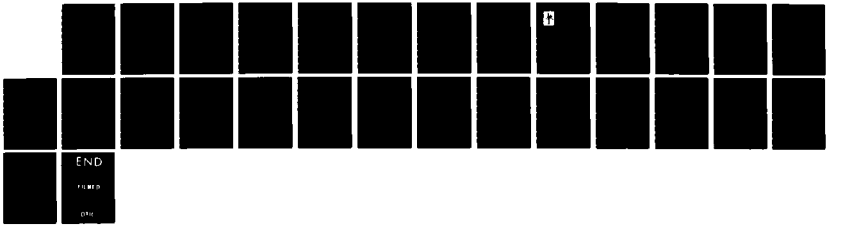
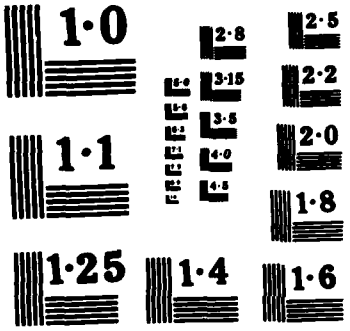
AD-A158 197 GENERATING AND GENERALIZING MODELS OF VISUAL OBJECTS (U) MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB J H CONNELL ET AL. JUL 85 AI-M-823 1/1

UNCLASSIFIED N00014-80-C-0505 F/G 6/4 NL





UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(2)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AIM 823	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Generating and Generalizing Models of Visual Objects		5. TYPE OF REPORT & PERIOD COVERED AI-Memo 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Jonathan H. Connell & Michael Brady		8. CONTRACT OR GRANT NUMBER(s) N00014-80-C-0505 N00014-77-C-0389
9. PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Intelligence Lab. 545 Technology Sq. Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
CONTROLLING OFFICE NAME AND ADDRESS lvanced Research Projects Agency 00 Wilson Blvd. lington, VA 22209		12. REPORT DATE July, 1985 13. NUMBER OF PAGES 24
MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) fice of Naval Research formation Systems lington, VA 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

DISTRIBUTION STATEMENT (of this Report)

stribution is Unlimited

This document has been approved for public release and sale; its distribution is unlimited.

DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

DTIC ELECTED AUG 2 1 1985 S D

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Vision
Learning
Shape Description
Representation of Shape

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We report on initial experiments with an implemented learning system whose inputs are images of two-dimensional shapes. The system first builds semantic network descriptions of shapes based on Brady's smoothed local symmetry representation. It learns shape models from them using a substantially modified version of Winston's ANALOGY program. A generalization of Gray coding enables the representation to be extended and also allows a sigle operation, called ablation, to achieve the effects of many standard induction heuristics. The

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

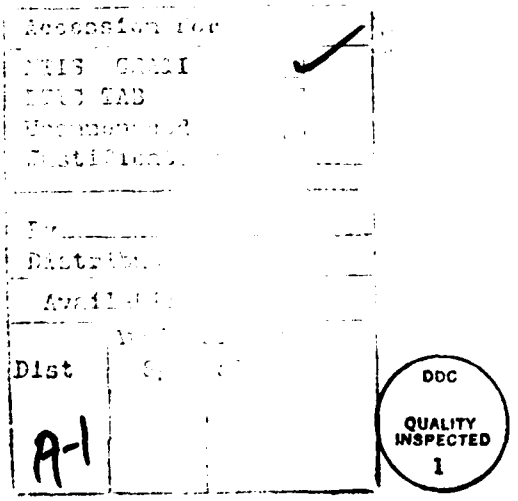
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD-A158 197

DTIC FILE COPY

20)

program can learn disjunctions, and can learn concepts using only positive examples. We discuss learnability and the pervasive importance of representatioal hierarchies.



MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo 823 July, 1985

Generating and Generalizing Models of Visual Objects

Jonathan H. Connell
Michael Brady

Abstract: We report on initial experiments with an implemented learning system whose inputs are images of two-dimensional shapes. The system first builds semantic network descriptions of shapes based on Brady's *smoothed local symmetry* representation. It learns shape models from them using a substantially modified version of Winston's *ANALOGY* program. A generalization of Gray coding enables the representation to be extended and also allows a single operation, called *ablation*, to achieve the effects of many standard induction heuristics. The program can learn disjunctions, and can learn concepts using only positive examples. We discuss learnability and the pervasive importance of representational hierarchies.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's Artificial Intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-80-C-0505, the Office of Naval Research under contract number N00014-77-C-0389, and the System Development Foundation.

1. Introduction

We report on initial experiments with an implemented system that learns descriptions of two-dimensional shapes. The inputs to the program are gray-scale images of real objects, such as tools, model airplanes, and model animals. The output of the program is a structured production rule that constitutes a procedure for recognising subsequent instances of a learned concept.

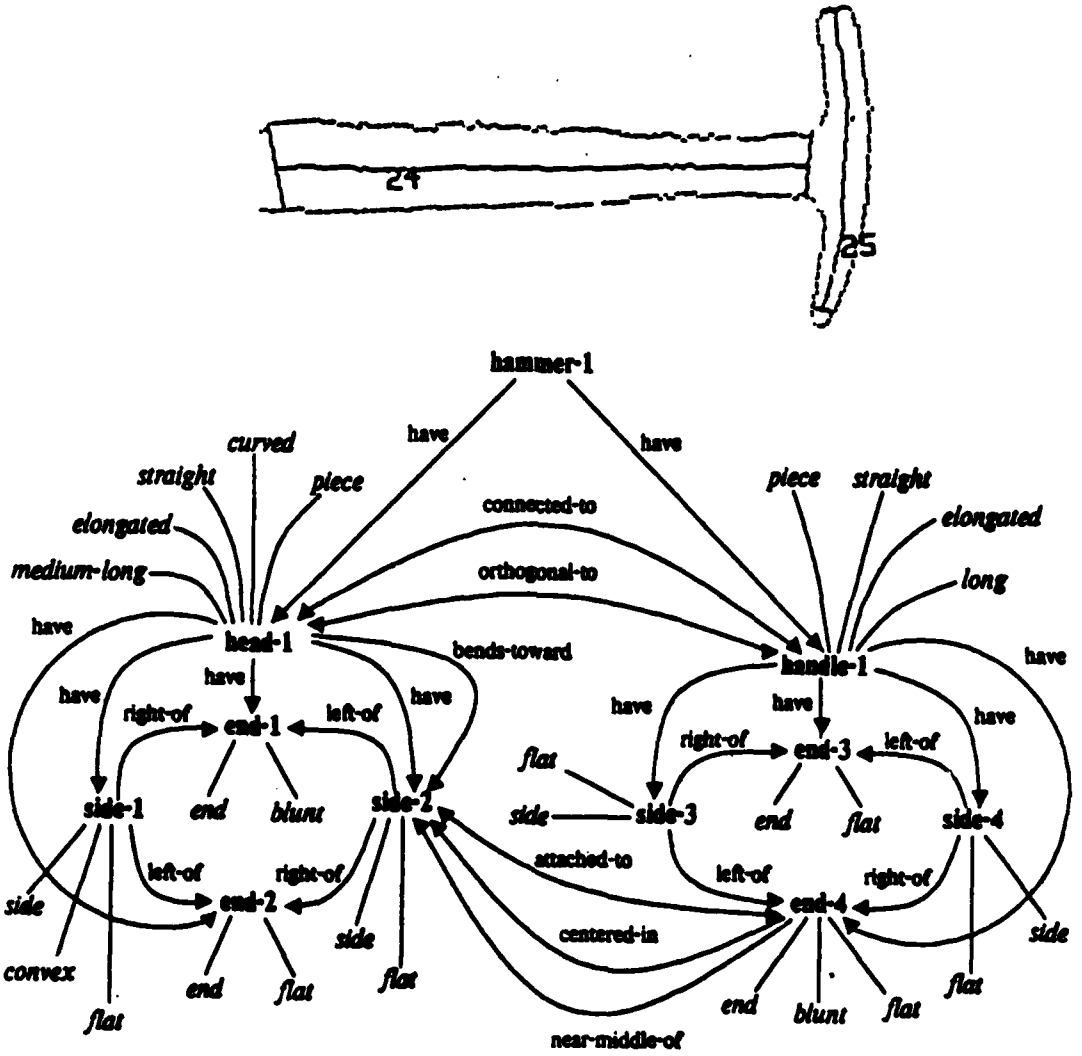


Figure 1. a. The main smoothed local symmetries extracted from the edge data of the tack hammer image. b. The semantic network that is computed from the information in a. The network generates 51 associative triples.

The system first builds semantic network descriptions of the imaged shape based on Brady's *smoothed local symmetry* representation [Brady and Asada 1984, Heide 1984, Bagley 1984, Fleck 1985, Connell 1985]. Figure 1a shows the major smoothed local symmetries computed from an image of a tack hammer, and Figure 1b shows the semantic

network that is computed from it by our program. Note that:

- The hammer shape is divided into two sub-shapes, one corresponding to the handle (*spine-24*) and one to the head (*spine-25*).
- The region occupied by the sub-shape is described symbolically. In Figure 1b, for example, the description of *spine-24* means that the handle of the tack hammer is *straight* and *elongated*. Similarly, the head of the hammer is *curved* and *elongated*. The spine (or axis) of the head *bends toward* the spine of the handle as one moves along it from left to right in the image. This distinguishes the hammer shape from the non-hammer shown in Figure 3a. The set of labels {*straight*, *curved*} for the curvature of the spine of the head illustrates our Gray coding approach to representing intervals of values in structures (see below).
- The bounding contour of each sub-shape (part of which may be hypothesized to complete the sub-shape) is also represented. The spine defines a local coordinate frame for the sub-shape, and portions of the bounding contour are labelled as *ends* or *sides* with respect to this frame. The *sides* and *ends* are described symbolically. For example, *side-1*, the top side of the head of the tack hammer, is determined to be *convex*, while *end-1*, the striking surface, is declared to be *blunt*.
- The join between the head and the handle is explicitly represented. There are two aspects of a join. First, as a relationship between regions, the spines of the head and handle are determined to be *connected-to* each other and to be *orthogonal*. Second, as a relationship between pieces of contour, the (the upper) *end-4* of the handle is *attached-to*, and, more specifically, is *near-middle-of* and *centered-in* the side *side-2* of the head.

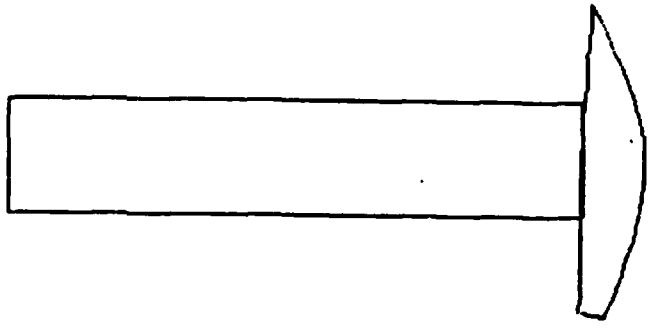


Figure 2. The symbolic descriptors in Figure 1a can be replaced by numerical values, and the resulting description treated as a graphics program. From this standpoint, the shape depicted in the Figure is the program's conception of the tack hammer shape.

The representation of the tack hammer shape depicted in Figure 1b is information-preserving in the following sense: each symbolic descriptor in the semantic network has an associated range of numerical values; it is possible to assign representative (for example, average) numerical values to the symbolic descriptors and interpret the resulting

description as a graphics program. In this way, the shape shown in Figure 2 more precisely reflects the program's conception of a tack hammer shape. Figure 3 also captures a sense of information preservation. Often, altering the descriptor of a single link in Figure 1b generates a shape that is a non-hammer. This has an important consequence for learning: the non-hammer shapes depicted in Figure 3 are "near-misses" but do not correspond to real objects; conversely, not every real object has near-misses.

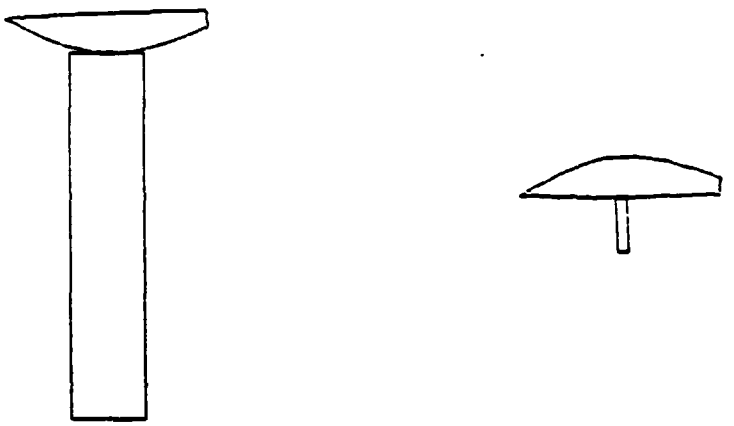


Figure 3. Varying individual descriptors in the semantic network in Figure 1b yields shapes that are non-hammers. a. The spine of the "head" does not *bend-toward* the spine of the handle. b. The spine of the handle is short.

The semantic network built by the program is transformed into a set of associative triples [Doyle and Katz 1985] and input to the learning component of the program. The tack hammer generates 51 associative triples. The learning program is a substantially modified version of Winston's *ANALOGY* program [Winston 1980, 1981, 1982; Winston, Binford, Katz, and Lowry 1984]. Connell [1985] discusses the differences in detail. In particular, the program is capable of learning concepts that are disjunctions. It can learn shape models using positive examples only. Figure 4b shows the concept *hammer* that is learned from the three positive instances shown in Figure 4a.

Figure 5a shows the gray-scale image of (a model of) a Boeing 747. Figure 5b shows several of the major spines found by Brady and Asada's smoothed local symmetries program. Each of the spines is a locus of local symmetries (as defined by Brady and Asada [1984]). Many of the spines correspond to perceptually important sub-shapes, such as portions of the wings, the engine pods, and the fuselage. The clutter in Figure 5b is because many smoothed local symmetries are *not* perceptually salient, for example that between a side of an elevator and a side of an engine pod. This illustrates the difficulty of dealing with real data rather than simulated data (more on this below). Figure 5c shows the spines that are extracted from Figure 5b and used to build the semantic network description of the airplane. The extraction process, detailed in [Connell 1985], uses two constraints: (i) a *good continuation* constraint for combining shapes that can be smoothly joined; (ii) a *uniqueness* constraint that seeks the most salient descriptor for a given region

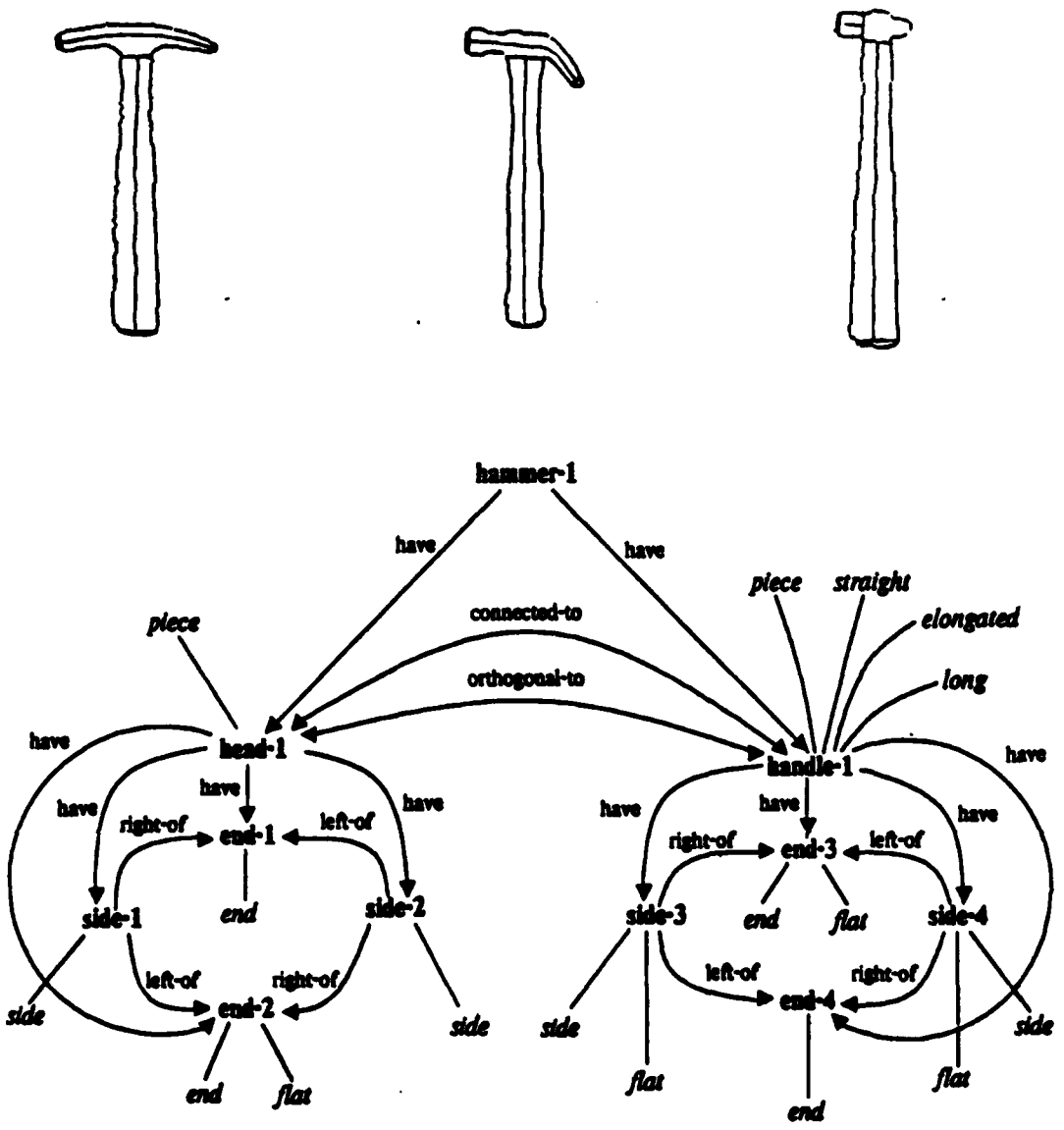
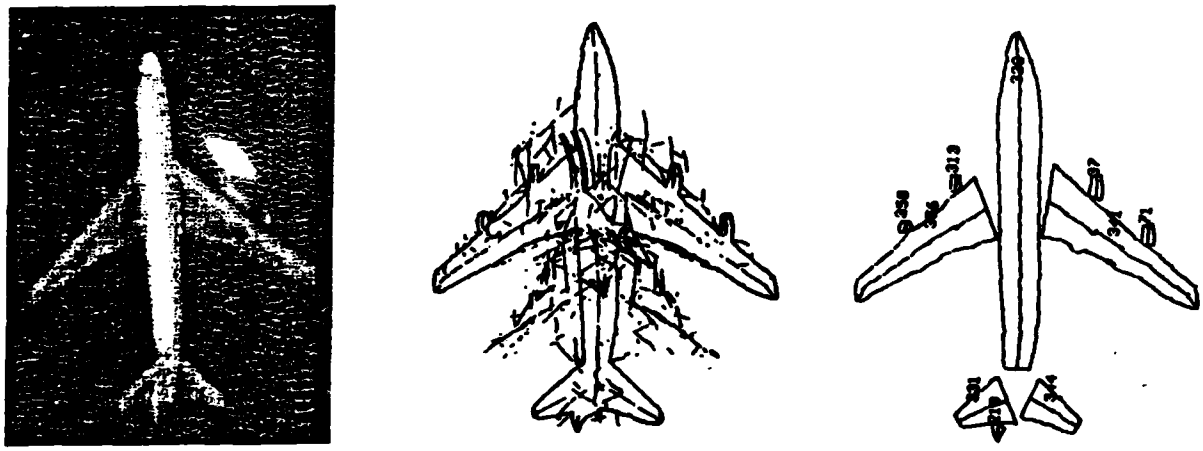


Figure 4. b. The concept *hammer* that is learned from the three positive instances shown in a.

or portion of contour. Part of the resulting semantic network is shown in Figure 5d. The B747 generates 239 associative triples.

The importance of learning from sensory information has been widely discussed in the Psychology literature (for example: [Hilgard and Bower 1966, Lindsay and Norman 1972]) and in Pattern Recognition [Duda and Hart 1973, Fu 1971, Fu and Tou 1974, Nilsson 1965]. Pattern recognition systems typically compute shape descriptions that are simply sets of features. Learning feature set descriptions mostly consists of modifying the weights accorded each feature for classification. For example, Michalski and Chilausky [1980] describe a program that learns descriptions of soybean diseases. Syntactic pattern recognition embraces the important idea of structural descriptions [Fu 1974, Gonsales and Thomason 1978] but has mostly been concerned with the properties of formal shape



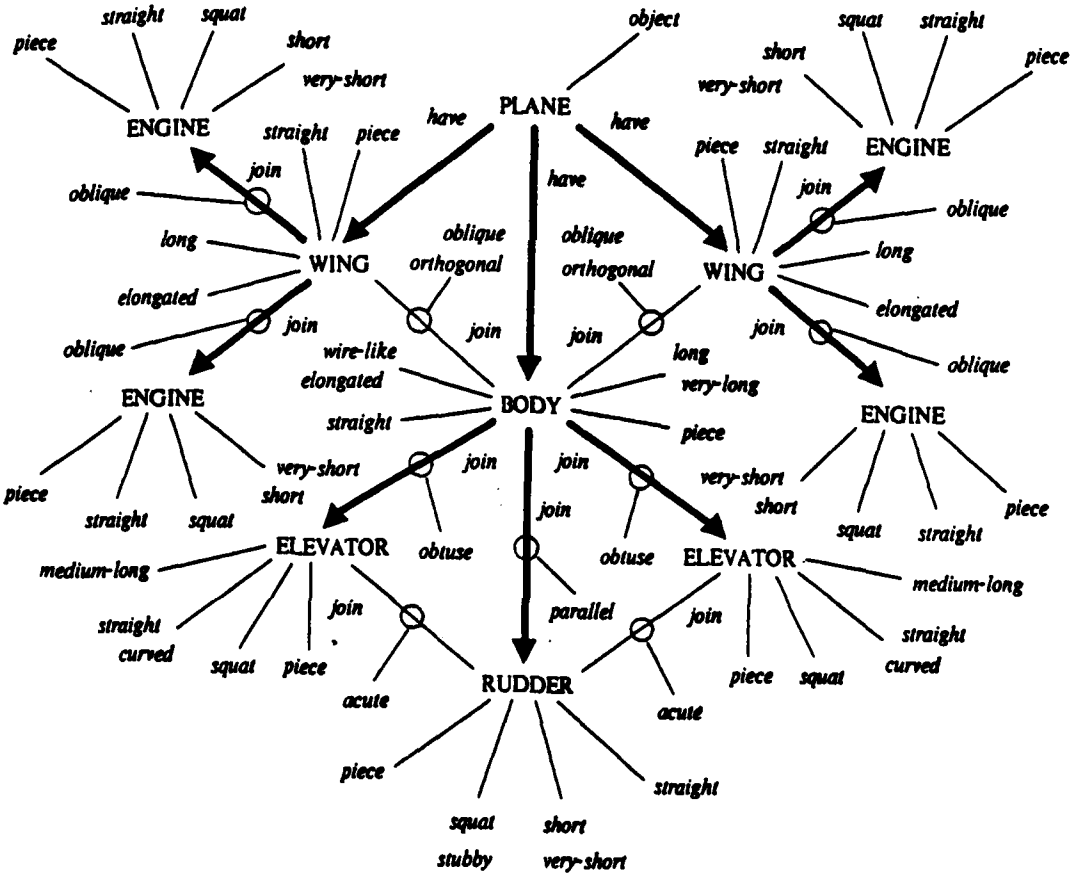


Figure 5. a. Image of a model B747. b. The smoothed local symmetries extracted from the edges of the image in a. c. The spines extracted from b. and used to build the semantic network description of the B747. d. Part of the semantic network of the B747. The full semantic network generates 239 associative triples.

grammars and with weak methods for recognition.

The novelty of our work is the ability to learn visual shape representations from real visual data. Previous work has not been based on real data because such data was unavailable or too complex and unstructured for existing learning algorithms. However, recent developments in edge-detection [Canny 1983] and middle-level (two-dimensional) vision [Brady and Asada 1984] have provided a solid base on which to build a robust vision system. Using this system we can generate shape descriptions in a form amenable to learning.

The descriptions of even simple shapes typically comprise between fifty and three hundred assertions, though various forms of abstraction keep this volume of data manageable. The semantic networks used in our experiments to date are not particularly huge. Winston's [1980, 1981, 1982] *ANALOGY* system, for example, has worked on larger semantic networks that capture some of the causal relations in stories. The novelty of the current work derives from trying to learn concepts using visual shape representations computed by a complete image understanding system. *Predicting* the assertions that will be generated from a new shape image is both tedious and error-prone. Discussing his work with *ANALOGY*, Winston [1984, page 438] observes "the theory was shaped to a great degree by experiments that would have been extraordinarily tedious to perform without the English interface, developed by Boris Katz, by means of which the experimental database was prepared, revised, and revised again. As Artificial Intelligence progresses beyond toy-world domains, it becomes obvious that databases prepared and accessed using English are necessary for research, not simply for research presentation." Tedium aside, the central issue is the *reality* of the data. Despite their extensive experience working with the shape program, the authors usually cannot accurately predict what description will be generated from a new shape image.

The next section of the paper describes, at some length, the representation of two-dimensional shapes that our program computes. This is not simply because we believe the representation to be of interest for image understanding in its own right, though, of course, we do. Rather, it is because, as Lenat and Brown [1984, page 291] state: "One of the most crucial requirements for a learning system ... is that of an adequate representation. The paradigm for machine learning to date has been limited to learning new expressions in some more or less well defined language." Section 3 relates shape to learning, and Section 4 outlines the learning component of the program. In Section 5, we sketch some work in progress on a project that relates function to form.

2. Representing Shape

2.1. Criteria

Redundancy is one of a number of criteria that have been proposed to evaluate a visual representation [Marr and Nishihara 1978, Brady 1983]. The objects of interest in an image are often only partly visible. This argues for a representation that contains useful redundancy. Smoothed local symmetries [Brady and Asada 1984, Heide 1984] are redundant in that they represent both the bounding contour of a shape and the region that

is subtended by the shape. If a section of the contour is noisy, or even if it is occluded, then the relevant information may be recoverable through the region description or vice versa. For instance, if one side of a subshape is convex and the other is concave we can infer that the spine of the region must be curved and that the direction of curvature is toward the concave side.

Here, we concentrate on another criterion that our representation satisfies: *stability versus sensitivity.* Images are noisy, so it is necessary to develop representations that are stable, in the sense of being invariant under localized changes such as image noise. However, tasks involving visual representations, for example inspection, often require that programs be sensitive to fine detail. A variety of techniques for simultaneously achieving stability and sensitivity have been proposed, each expressing some aspect of hierarchical description. The underlying idea is that gross levels of a hierarchy provide a stable base for the representation, while finer levels increase sensitivity. Coping with highly structured representations is a key issue for learning, since the vast majority of programs that learn from sensory information represent shapes by a flat set of assertions or by feature sets.

2.2. Visual hierarchies

Several representational hierarchies have been developed in Computer Vision. In this subsection, we describe those that are used in our program:

• Numeric values and symbolic descriptors

Specifying a shape parameter of interest, say a measure of the elongation of a shape, by a numerical value is sensitive, but highly unstable. Symbolic names that correspond to an interval of numeric values are (usually) more stable but less sensitive. Heide [1984] showed that the algebraic relations developed by Brady and Asada [1984] for smoothed local symmetries could be used to compute a useful set of perceptual descriptors for a shape: elongation; increase in the cross section of a shape, together with the rate of increase; the summarized curvature of an arc between significant curvature changes (convexity, concavity, or straightness of the spine of a region); as well as a descriptor of a primitive shape type (beak, wedge, etc.). Heide demonstrated the use of numeric and symbolic descriptions of these parameters in describing several dozen shapes.

We have built upon Heide's work in several ways, most especially in describing subshape joins, as discussed below. Also, our representation employs symbolic descriptors that have overlapping ranges. For example, an end which is determined to be on the borderline between blunt and sharp is declared to be both blunt and sharp. Overlaps like this help to combat the quantization error introduced by encoding a continuous range as a set of discrete symbolic values. We show later how overlapping ranges are implemented using a variant on the technique of Gray coding [Hamming 1980, pages 96ff].

• Scale space filtering

Signals are often perceived as having a different structure at different spatial scales. For example, a saw blade may be considered straight at a coarse scale, yet sawtoothed at a finer scale. In the *Curvature Primal Sketch* [Asada and Brady 1984, and the scale

space references therein], which forms the basis for contour description in *smoothed local symmetries*, the contour of a shape is filtered with Gaussians of increasing width; then the extrema of the smoothed curvature and its first derivative are located and matched to multiscale symbolic models. Descriptors include: *end, corner, crank, bump*, and *smooth join*. This information will eventually be incorporated into our representation of objects.

- **A-kind-of hierarchy**

Family hierarchies are ubiquitous, and apply as much to visual shape representations as to the more cognitive situations in which they were developed in Artificial Intelligence. *ACRONYM* represents the fact that the sets of B747-SPs, B747s, wide-bodied jets, jets, and aircraft, are ordered by subset inclusion. Similarly, a claw hammer is a-kind-of framing hammer, which is a-kind-of hammer. In general, a subset hierarchy is a partially-ordered set, but not a tree. From the domain of tools, for example, a shingle ax is both a-kind-of ax, and a-kind-of hammer.

- **Structural approximations to shapes**

Marr and Nishihara [1978] proposed summarizing the lesser subparts of an object, leaving them unspecified until they are needed. For example, all airplanes have a fuselage, with pairs of symmetrically attached wings and elevators. Upon closer examination, a wing of a B747 has two attached engine pods, a DC10 has one, and an L1011 none. Suppressing mention of the engine subshapes, as well as summarizing the parameters that describe the shapes of the wings and fuselage, enables the descriptions of the three airplanes to closely match each other.

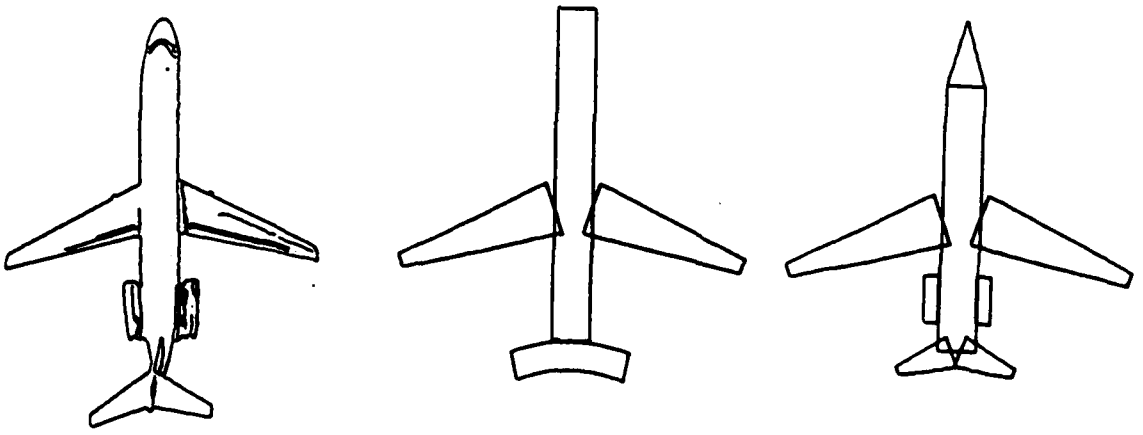


Figure 6. a. The engine pods of a B727 are directly attached to the fuselage. b. The coarsest description mentions only the fuselage, wings, and tailpart, allowing the shape to be recognized as an airplane. c. Refining the description of the fuselage involves mentioning the attached engine pods and the sub-parts of the fuselage spine, such as the nose section.

The Marr-Nishihara proposal tends (heuristically) to relate large scale geometric structure to gross functional use. Sub-shapes hierarchically organized by attachment and by size often provide useful access to functional descriptions. The *ACRONYM* system [Brooks 1981, Brooks and Binford 1980] can use the context provided by matching models at the gross level to reason about smaller, locally ambiguous, image fragments.

In general, larger subshapes tend to determine gross categorization, and so they tend to appear higher in the structural hierarchy. Conversely, smaller subshapes tend to allow finer discrimination and to occur lower in the hierarchy. These two statements embody a *big fleas have little fleas* heuristic about articulated shapes: a sub-shape tends to be (recursively) smaller than the sub-shape of which it is a part. If only geometric information is available, the relative sizes of sub-shapes, and the connection topology are the best bet for developing a hierarchical representation. For example, although the engine pods of a B727 are attached directly to the fuselage, size dictates that the coarsest description mentions only the fuselage, wings, and tail section (Figure 6). Refining the description of fuselage involves mentioning the attached engine pods and the sub-parts of the fuselage spine, such as the nose section.

Small does not mean unimportant. Often, the smaller subparts of a tool determine its specific function. For example, deciding whether a tool is an awl, a gimlet, or a screw driver of the Philips type involves looking closely at the end of the blade; the relatively localized *context* of the business end of the blade is established by the grosser levels of the hierarchy, where it is recognized (for example) that the tool is not a hammer or wrench.

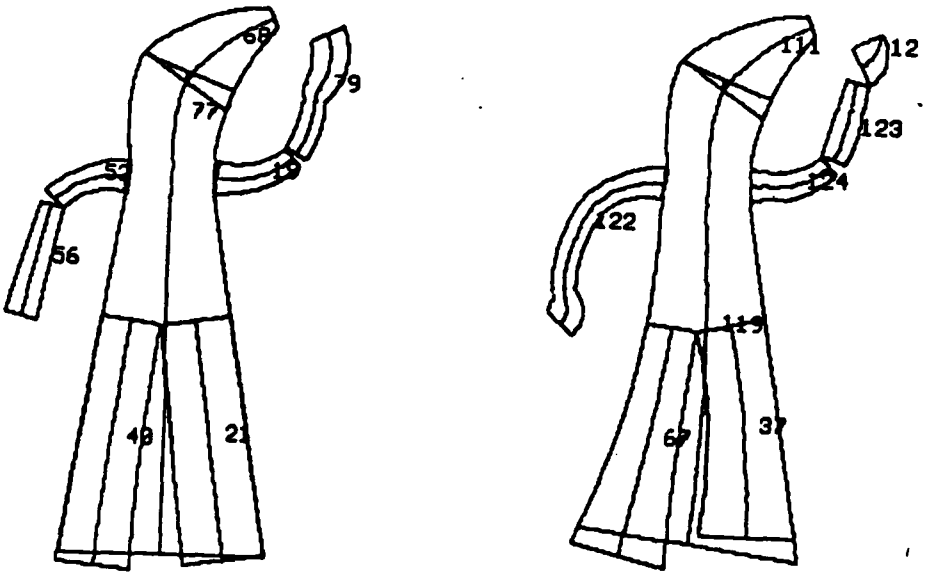


Figure 7. Scale does not correlate with abstraction. a. The sub-shapes extracted from the coarsely smoothed contour of Gumby. b. The sub-shapes extracted in less smoothed version.

It is tempting to equate Marr and Nishihara's proposal with scale-space filtering. Figure 7 shows that they are quite different. Figure 7a shows the sub-shapes extracted from an image of the cartoon character Gumby after its contour has been smoothed with

a relatively wide Gaussian filter. Both arms are separated at the elbow, and neither hand can be seen. This is similar to an intermediate representation suggested by Marr and Nishihara. Figure 7b shows the sub-shapes extracted at a finer scale. One arm is again divided at the elbow, and the hand has been made explicit, in further accordance with Marr and Nishihara's proposal. However, the other arm is not divided at the elbow as the arm no longer has a significant change in curvature.

2.3. Describing sub-shapes and joins

The representational hierarchy requires that sub-shapes, and sub-shapes of sub-shapes, be isolated, and the way in which they are joined together be described. Brady and Asada [1984] proposed a method for isolating sub-shapes, but Heide [1984] showed several counter-examples. In turn, we have found that Heide's method is inadequate. We have developed a method that appears to be efficient and effective, and which satisfies the representational criterion of *redundancy*. It involves both regions and contours: regions are grouped together if their spines smoothly extend *and* their contours smoothly extend. For example, the portions of fuselage in front of and behind the wings of the B747 in Figure 4 are joined, but the handle and blade of a screw driver are perceived as separate pieces since there is a width discontinuity generating contour steps. In terms of the mathematical analysis in [Brady and Asada 1984], a sub-shape is defined as maximal with respect to smooth variations in the defining parameters. Joins between sub-shapes are determined by examining the spines of the regions and the adjacency of the contour segments.

The following descriptors are computed for sub-shapes:

- whether the region is *filled* or is a *cut-out*;
- whether the (relative) length of the region is *short*, *medium*, *long*, or *very long*;
- whether the elongation of the region is *squat*, *elongated*, or *very elongated*;
- whether the spine of the region is *straight*, *curved*, or *very curved*;
- if the spine is curved, which of its sides does it *bend toward*; (In Figure 1, a side is *right-of* an end if it is on the right when you stand at the end specified and view along the spine.)
- whether the ends of the sub-shape are *flat*, *blunt*, or *sharp*;
- whether the sides are *flat*, *convex*, *concave*, *very convex*, or *very concave*.

Many of the descriptors of spines and contour segments given above are illustrated in Figure 1b. For further examples, and formal definitions, see [Connell 1985]. Deciding that a portion of the contour of a shape is an *end* or a *side* first requires choosing a spine to describe the region. In the smoothed local symmetries representation, there may be more than one spine that covers a given region, hence more than one description involving different perceived pairs of ends and sides.

The discussion of Figure 1b observed that a join between sub-shapes usually involves a relationship between the corresponding spines and between the corresponding contour

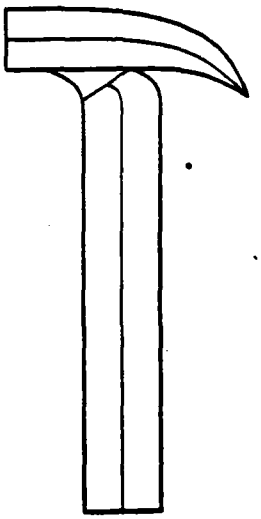


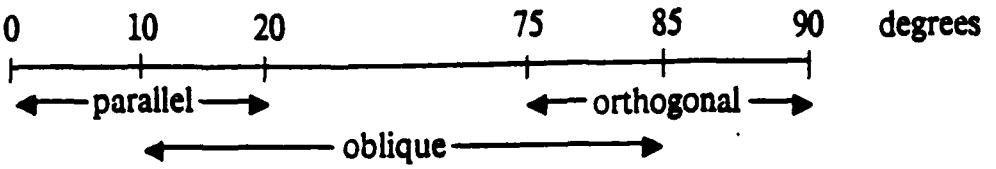
Figure 8. The angle between two joined regions is determined from the angle between the spines. Here the regions are *orthogonal*.

fragments. Our descriptions of joins contain the following information:

- which region is connected to which region;
- which edge is joined to which edge, where an edge is either an end or a side. For example, the upper end of the handle of a hammer joins the lower side of the head;
- what the angle between the joined regions is. This is computed from the spines rather than from the contour to give a more global estimate. In Figure 8, the stem of the T-shape is *orthogonal* to the cross-bar. Note that estimating the angle between the joined parts from the curvature change along the contour would give different results. The spines of the joined regions are parallel, perpendicular, or oblique (Figure 9a gives details);
- if the end of one shape Σ_1 is joined to a side of another shape Σ_2, which end of Σ_2 does the spine of Σ_1 lean toward (make an acute angle with);
- if the end of one shape Σ_1 is joined to a side of another shape Σ_2, is the join near an end of Σ_2 (if so, which end), is it near the middle, or is it centered (Figure 9b gives details).

Many of the join descriptors are illustrated in Figure 1b. Further examples and formal definitions can be found in [Connell 1985]. Few previous representations of shape have described sub-shape joins. For example, *ACRONYM* specified the coordinate transformation between two joined pieces, but did not make the join explicit or describe it symbolically.

a. Angle quantization



b. Join locations

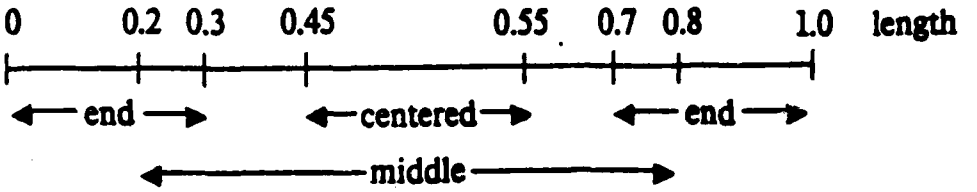


Figure 9. a. The initial quantization of the angle between the spines at a join. b. The initial quantization of the placement of a join in a side-end join.

3. Shape and Learning

3.1. Explicitness and Extensibility

We noted in the previous section that a vision program that deals with shape needs to maintain several different representational hierarchies, including number-symbol, multiple scales, a-kind-of, and structural approximation. It follows that a system that is to learn such a visual shape representation must *respect*, and possibly *add to*, each of these hierarchies. Conversely, the need to learn shapes at every point in the hierarchy imposes certain *learnability* demands on the vision system that generates the representation.

Our program respects the hierachical nature of the representation by using *structure mapping* as a generic process for learning and recognition. This is illustrated by the figure above that shows the hammer model being learned. To support structure mapping, we offer the following principle: *syntactic distance should reflect semantic distance.* In our case, it means that structures that look similar should give rise to similar decriptions. In programming there are two extreme strategies for encoding the additional knowledge. According to the (strictly) procedural strategy, the knowledge is encoded as a set of complex procedures that implicitly know how to interpret a simple data structure. In the declarative strategy, knowledge is made explicit in complex data structures that are

manipulated by very simple programs.* The semantic networks shown earlier should suggest (correctly) that we have adopted a declarative strategy.

However, no shape representation is static. In the course of our work, we have needed to modify our representation in two main ways to enable our program to learn new concepts. First, it has been necessary to add new kinds of descriptors and data types. For example, we first developed the descriptors enumerated in the last section and illustrated in Figure 1b. Then we found it useful to view *region* as a new data-type that could be compared to find out how similar they are. Similarly, we have begun to add functional information, and need to compare different functions.

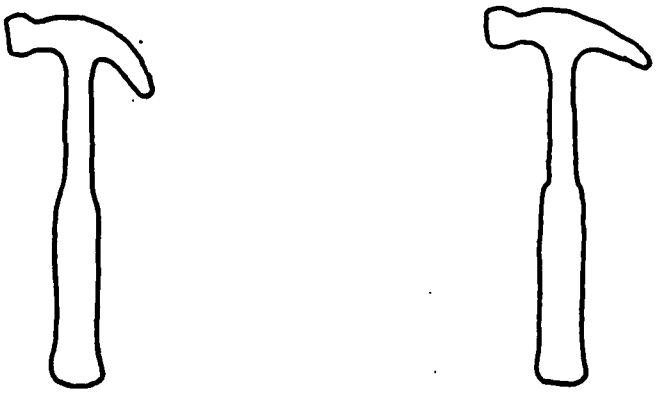


Figure 10. Silhouettes of a claw hammer (left) and a rip hammer (right). The apparently slight difference in curvature of the peins of the two hammers has substantial functional significance. Most people have never heard of rip hammers. Typically, they are recognized as claw hammers.

Learning a new shape may also expose limitations in one's representation. If the distinction required to distinguish between two objects cannot be articulated, the concept cannot be learned. How can a program (or person) learn to distinguish between a claw hammer and a rip hammer (or between a claw hammer and an electrician's hammer [Brady, Agre, Braunegg, and Connell, 1984])? Most people have never heard of rip hammers, and typically recognize a rip hammer as a claw hammer. They do not know about rip hammers because *they have never needed to* use one. Construction workers use them regularly. Expertise calls for extra fine judgement, requiring increasingly refined scales of description. From Figure 10, the program needs to learn that the pein of a claw hammer is more strongly curved than that of a rip hammer; knowing that, the program may then wonder why that is the case, and it may suspect a slight variation in function (guessing the function of each hammer is left as an exercise for the reader). * A representation that only distinguishes between curved and straight sub-shapes, such as peins, will be capable of representing the difference between a mason's hammer and a claw hammer, but not that between a claw and a rip.

* The qualifier "typically" should be applied to "complex" and "simple".

* A related question asks whether there is any functional significance in the curvature of the pein of a claw hammer. Answering this question gives a hint to the shape of the rip hammer.

This is a (typical) case in which the representational inadequacy reflects an *insufficient resolution* hassle [Brady, Agre, Braunegg, and Connell, 1984]. To correct such an inadequacy the program must first understand what is wrong. Once the nature of the hassle is determined, the program can modify its representation by incorporating the appropriate additional knowledge. Our approach has been to simply tell it: "See, the pein of a claw hammer is much more curved than that of the rip hammer", suggesting that the curvature of the spine of the pein is described with insufficient resolution. Refining the representation involves using a more detailed quantization of the continuous curvature scale. In general, a learning program may require changes to be made at arbitrary depths in the visual representation.

3.2. Gray coding

The syntax of our representation is adapted from the idea of Gray numbers used in digital communication [Hamming 1980, pages 96ff; Davison and Gray 1976]. Toggling a single bit in a Gray-number only changes the value it represents by one (for example, one might be coded as 01, two as 11, and three as 10). In this way, semantic distance (the mathematical difference between encoded numbers) corresponds to (syntactic) Hamming distance. In digital communication, this property means that the effect of losing any one bit is uniformly non-catastrophic. In shape representation, it means *all* the knowledge in the representation can be made explicit; the similarity metric does not have to be hidden away in the control structure of the system.

While the Gray coding of numbers was originally developed for digital communication, our interest in Gray coding is rather different. We have extended this technique to the different data types in our representation to yield a *uniform closeness metric* that can be simply weakened or refined. A novelty of our approach has been to show how:

- Gray coding can be extended to continuous intervals and data structures, allowing new data types to be added easily to the representation;
- finer distinctions can be made by increasing the number of binary predicates used to Gray code the interval;
- a *single* operation called *ablation* can be applied to Gray coded structures to achieve the effect of the half dozen or so induction heuristics developed by Michalski and others.

Consider first encoding intervals, which may either be discrete or continuous. For example, our representation maintains the discrete set

$$\{flat, convex, concave, very\ convex, very\ concave\}$$

of symbols for the curvature of a portion of contour. However, we also maintain a continuum of numerical values for the curvature. Intervals and linear descriptors can be encoded by using overlapping ranges. Equivalently, as shown in Figure 11, we create a set of overlapping binary predicates a through j whose ranges cover the interval. For each range we have a boolean variable which is true if the value being encoded falls in

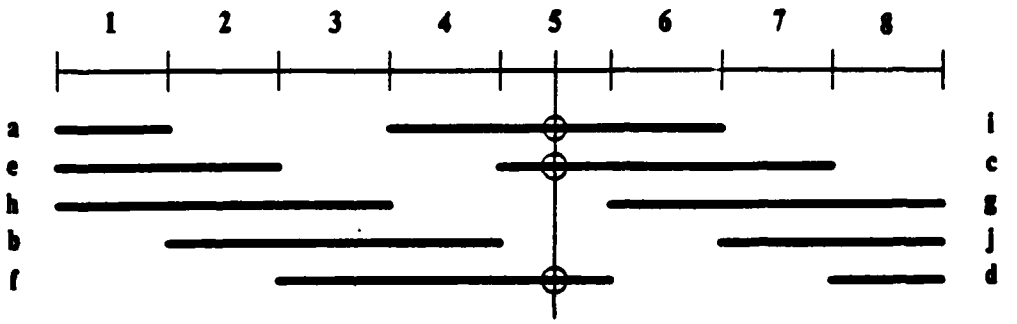


Figure 11. Gray coding extended to intervals. The set $\{c, f, i\}$ represents the value five.

its range and false otherwise. This means a particular value is encoded by the set of intervals it is in: four is encoded as $\{b, f, i\}$, five as $\{c, f, i\}$, and six as $\{c, g, i\}$.

We can use counting to determine how close two values are. Notice that the representations of four and five differ by one, five and six by one, and four and six by two. The *uniform closeness metric* property of Gray coding guarantees that sets that differ by a small amount correspond to nearly equal values. Also, as generalization is intimately related to similarity, we can express generalizations in the same language. For instance, a generalization which is the interval $[4 \ldots 5]$ can be represented by the set $\{f, i\}$. Note that this set subsumes the sets corresponding to four and five.*

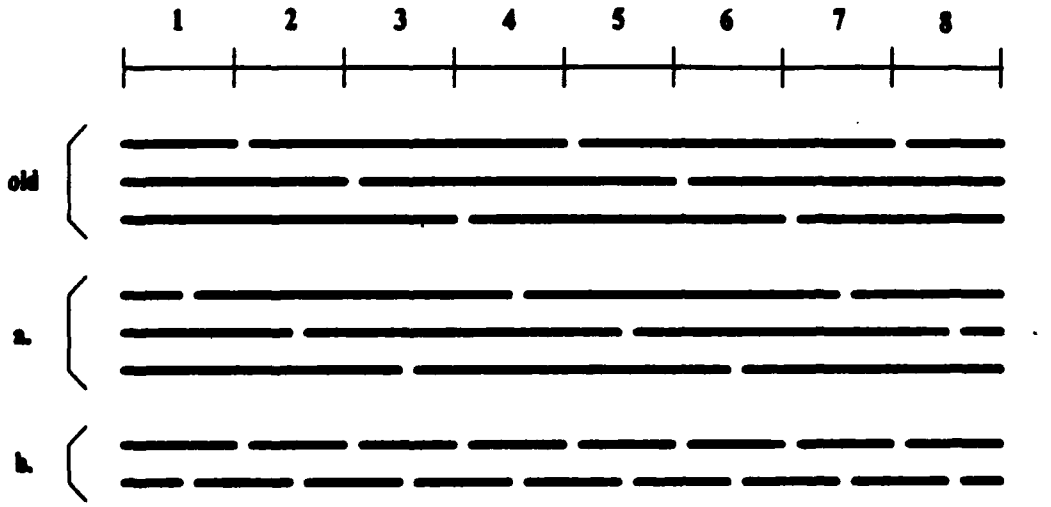


Figure 12. Increasing the resolution of a Gray-coded interval. a. More predicates of the same width as the originals are added. b. New narrower predicates are added.

* It has been pointed out to us that a similar encoding technique has been used in phonology [Kenstowicz and Kisseberth 1979, chapter 7; Chomsky and Halle 1968].

A further property of Gray-coding is that it is simple to increase the resolution at which a quantity is represented. Recall that learning to distinguish a rip hammer from a claw hammer involved a finer discrimination in the curvature of the pein. Suppose we initially divide an interval into eight discrete ranges and create the corresponding Gray-code predicates. Later we find that eight divisions are not enough, but that we really need something like sixteen divisions. The granularity of the representation can be reduced by simply adding more predicates. One way to do this is to add predicates corresponding to ranges which are same width as the original set but offset by half a unit as shown in Figure 12a. Another possibility is to add ranges which are smaller than the original as in Figure 12b. In either case a particular number will now be represented by a set of four predicates instead of a set of three. The key point is that the mechanisms that worked before continue to work on the refined representations.

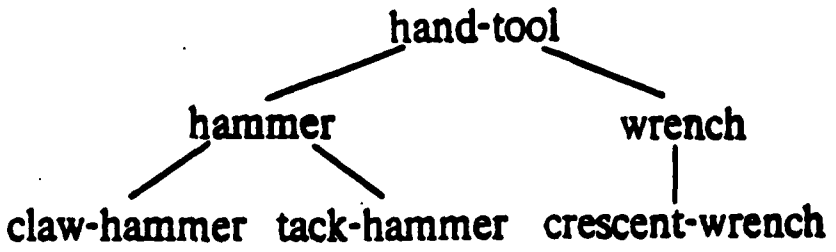


Figure 13. Part of an a-kind-of hierarchy. The set {*tool*, *wrench*, *crescent*}. represents a crescent wrench.

Gray coding can be extended to trees, or any other data type. Consider the family tree shown in Figure 13a. A tack-hammer is a-kind-of hammer, which is a-kind-of hand-tool. We Gray code a node n in the tree by the set of nodes from n to the root. For example, the claw hammer is encoded as {*claw*, *hammer*, *tool*}, the tack hammer as {*tack*, *hammer*, *tool*}, and the crescent wrench as {*crescent*, *wrench*, *tool*}. Counting, as for intervals, we find that the claw hammer is most similar to the tack hammer. Furthermore, we can make generalizations such as {*tool*, *hammer*} and, as before, refine the representation by adding more predicates. The same set of procedures that worked on intervals now also works on trees.

4. The Learning System

4.1. Background

Our approach to learning is based on Winston's work on analogy [Winston 1984]. Indeed, the experiments reported here began when we interfaced *ANALOGY* to the Smoothed Local Symmetry representation and image understanding system. Structure matching

plays a central role in *ANALOGY*, the structures being representations of the sort described previously. Like *ANALOGY*, our program can generate a set of rules that amount to a procedural description of how to recognize a shape. Rules can be modified by censors [Winston 1984, page 432], and censors can block other censors [Winston 1984, page 435]. This section describes some of the major differences (see [Connell 1985] for further details), including the complex local matching scheme used by the program, and the way in which the *drop condition* heuristic has been generalized to intervals and structures.

Mostly, the program learns concepts using positive examples only (so do the programs of Anderson [1977], Berwick [1985], Vere [1977], and Winston [1984]). We emphasise this because many of the concepts, particularly generic concepts, that we wish to learn do not have "near misses". Whereas alternative sub-categories, such as B747 and L1011 provide near-misses for each other, it strains one to think of a real object that is a near miss of a major category, such as airplane. Even when such near misses exist, people seem to be able to learn the concept without seeing them.

Our initial goal is to learn shape models cast in the representation described in the previous section. We need to tolerate variability within a class of shapes (hammers, say), yet discover subclasses (say, claw hammers, rip hammers, and shingle axes). The hierarchical structure of the shape representation facilitates this. However, the work reported here also forms part of the *Mechanic's Mate* project [Brady, Agre, Braunegg, and Connell 1984; Agre 1985]. Eventually, the *Mechanic's Mate* will have to learn about the non-geometric properties of objects as well. These properties include: weight, material type, and the processes that use them.

4.2. Ablation

The commonest form of inductive generalization used to learn concepts from positive examples is the *drop condition* heuristic [Dietterich and Michalski 1981, Winston 1984, page 398]. In our program, *drop condition* has been generalized to a process we call *ablation*, which works because we use Gray coding. Through careful design of the shape representation, ablation achieves the effect of *drop condition*, and the other induction heuristics listed by Dietterich and Michalski. For example, the effects of the *forbid link* and *require link* heuristics associated with "near misses" can be achieved without special machinery for dealing with near misses, though they require negative examples.

The idea behind the ablation heuristic is that if two things belong to the same class then the differences between them must be irrelevant. Accordingly, when we have a partial model of a concept and receive a new example, we modify the model by deleting all the differences between it and the example. This can be seen by comparing Figure 1b with Figure 4b. Notice that the network in Figure 4 puts very little constraint on the size or shape of the head. This is because the shapes of the heads in the examples that are presented to the learning program vary widely. For instance, the heads of the first and third hammer are straight while the head of the second hammer is curved. Note also that the manner in which the handle joins the head is only loosely specified. This is because the handle is joined to the side of the head in the first two examples but to the end of the head in the third example. Of course, if there are many substantial differences between the model and the new example, applying ablation can lead to overgeneralization. In

such cases, our program generates a disjunctive concept consisting of the model and new example. We discuss disjunction further below.

Ablation only works because the syntax of our representation reflects its semantics in a very simple way. As mentioned before, the program measures the similarity of two nodes comparing their representations: the more properties they have in common, the more similar they are. If the value associated with a node is a binary predicate (for example, a sub-shape is either a cut-out or filled in), then similarity means that both nodes have the same value. This suggests that we can make a generalization based on two examples by simply leaving out all the things that don't match. Since counting is used to measure how similar two representations are, it is easy to tell what to leave out. Since we use Gray coding, we are *guaranteed* that leaving out facts creates a reasonable generalization. Obviously this technique achieves *drop condition*; in different circumstances, ablation achieves the effect of other induction heuristics as well. In the last section we showed how Gray coding could be used to create generalizations by removing items from the descriptor set. In the examples involving numeric values and trees, ablation performs the *close interval* and *climb tree* operations, respectively.

The statement that the learning algorithm is based on applying ablation to suitably crafted representations is a highly simplified explanation of how it actually works. The matching involved is not graph isomorphism nor does it merely involve counting the number of required features an object has. Rather it is a complex *local matching* scheme. Consider using the semantic net shown in Figure 5 as the model for the airplane concept. For an object to match this model, at the top level it must have three pieces which look similar to the three in the model. A piece of the example is similar to the wing model if, first of all, it has the shape specified in the network and, second, it has two things which look like engines attached to it. Suppose that a certain piece has the right shape for a wing but has only one engine attached to it. At the level of the wing model the program notices that there is a discrepancy, yet it judges that the piece is still close enough to the description to be called a wing. When the top level of the matcher asks if the piece in question looks like a wing the answer is "yes"; no mention is made of the fact that the wing is missing an engine. The difference only matters locally and is isolated from the higher levels of matching. This does not mean the program forgets that there was a difference: it needs to remember this at least to perform ablation. Rather, it means that the program's judgement of the similarity of two objects is based on a number of very simple *local* decisions.

4.3. Disjunctions

Imagine that the program is shown a positive example that is substantially different from its current model. As mentioned before, altering the model by the usual induction heuristics typically leads to gross over-generalization. This, in turn, runs counter to what Winston [1984, page 401] has dubbed *Martin's law*, namely: learning should proceed in small steps. Winston overcame this problem by relying on a teacher to provide near misses. Examples that differed too much were unusable. While this certainly works, it puts much of the burden of learning on the teacher. Therefore, instead of disregarding

such an example, our program assumes that the concept being taught is a disjunction and creates a new, separate model based on the example.

In some cases, the disjunction will be replaced by a single model as positive examples are taught that are intermediate to the arms of the disjunction. For example, suppose that the first example of a hammer shown to the program is a claw hammer, and that the second is a sledge hammer. The program will create a disjunction as its concept of hammer, but it will be consolidated into a single model once it has seen such examples as a mallet and ballpein hammer. It seems that disjunction can serve a purpose similar to Winston's *near-miss-group*. His idea is to form the intersection of the differences between a set of "far" misses and an evolving concept. In this way, a set of far miss examples can serve a role similar to a single, focussed, near miss. Similarly, the two arms of the disjunction can serve as a single positive example.

In general, however, representing a concept as a disjunction suggests that the representation is inappropriate. This is regrettable, but often unavoidable: learning often involves groping toward the right representation. Stepping out of the tool domain, consider learning the concept "mammal". Initially, one might be impressed by the observation that all the familiar legged creatures like people, horses, and cats are mammals, while snakes and fish are reptiles. One is then told that a whale is a mammal. This leads us to propose that a mammal is either a "familiar legged beast" or a whale, considered as a rather mysterious exception. Discovering later that all reptiles lay eggs, that no mammal does, and that egg laying is something to do with reproduction, might suggest reproduction as the correct representation space. However, for some concepts, such as "bird", there does not seem to be a correct representation space. While the *Mechanic's Mate* will know nothing of reproduction, it will need to discover that functionality, not geometry, is the appropriate representation space to unify wrenches and pliers.

Even though the program only generalizes a concept using an example that is structurally similar, it is sometimes deceived, and must recover from over-generalization. We follow Winston [1984] and provide censors that over-ride the offending rule. Censors can be generalized and there can be disjunctive censors, in fact this is the usual case. Since censors can be generalized they also have the possibility of being over-generalized. This is countered by putting censors on the censors. In general, a concept is not represented by a single model but by a group of models. There can be several positive models corresponding to the arms of a disjunction as well as several negative non-models summarizing the exceptions to the other models.

4.4. Sub-classes

We noted earlier that our representation contained many a-kind-of hierarchies: a B747-SP is a-kind-of B747, which is a-kind-of wide-bodied jet, which is a-kind-of jet, etc. Such hierarchies are important because they can speed matching in a huge database of models. Suppose a claw hammer example has been matched to the hammer model, establishing a correspondence between a part of the example and the head of the generic hammer model. The a-kind-of link from the head of the claw hammer model to the head of the hammer model suggests that the example part corresponds to the head of the claw hammer model. So matching does not need to start anew as we move down the hierarchy.

Since the subclasses that form the a-kind-of hierarchy are an important part of the shape representation, they should be learnable. However, learning subclasses is dangerous insofar as there is the danger of combinatorial explosion if we form all possible subclasses. One way around this is to use an algorithm like those in feature-based pattern recognition systems. These systems learn subclasses as clusters in feature space, where clusters are sets that are dense with respect to the Euclidean metric. The Gray coding metric described previously respects the hierarchies in our representation, suggesting that we may generalize feature space clustering to learn subclasses. For instance, the representations of a 12-ounce claw hammer and a 16-ounce claw hammer are much closer to each other than to the representation of either a tack hammer or an electrician's hammer. Noticing this, the program might propose that the two claw hammers constitute an interesting subclass of hammers. This is one focus of our current work.

5. Reasoning between function and form

The goals of our research are not limited to learning structural descriptions. As mentioned before, this work is one part of the *Mechanic's Mate* project [Brady, Agre, Braunegg, and Connell 1984; Agre 1985], which is intended to assist a handyman in generic assembly and construction tasks. We are particularly interested in solutions to construction problems that might be considered *innovative* in that they adapt tools designed for other purposes or use the expected tool in a novel way. The primary interest of this project is to understand the interplay between planning and reasoning that involves tools and fasteners and the representations of the objects' shapes. In turn, an important aspect of such reasoning is the relation between form and function.

Pioneering work in relating form and function was carried out by Freeman and Newell [1971]. Davis [1984] also exploited form and function in troubleshooting digital circuits. Recently, a whole issue of *Artificial Intelligence* was devoted to reasoning between form and function. Our approach, however, most closely resembles the system discussed by Winston, Binford, Katz, and Lowry [1984]. Unfortunately, they note that "At the moment, the physical description is given in English, bypassing vision. The description is couched in the generalized cylinder vocabulary of *ACRONYM*" [Winston, Binford, Katz, and Lowry 1984, page 121]. This influenced us to begin the present study, which employs real visual data rather than verbal descriptions.

The basic idea is that, instead of learning that a certain geometric structure is named a hammer, we learn that *something which has a graspable portion and a striking surface* can be *used as* a hammer. These two functional concepts are then defined geometrically in terms of the shape representation. The graspable portion should have a description like the hammer handle in Figure 1b: it should have a spine that is *straight* and *elongated*, and *sides* that are only slightly curved. The striking surface should be an *end* of a sub-shape (perhaps the handle sub-shape) that is *blunt* and that is parallel to the spine of the handle.*

Now, faced with a hammering task, but no hammer, we can try matching the hammer shape description to that of any available tool. A close match in geometric form might

* The requirement of parallelism would need to be relaxed for a modern, ergonomic, hammer.

suggest that the matched tool be adapted for use as a hammer by grasping the portion that is matched to the hammer handle and striking with the portion matched to the hammer striking surface. Our program suggests that a screwdriver provides a good match: identifying the blade of a screwdriver with the handle of the hammer, and the (assumed flat) side of the screwdriver handle as the striking surface of the head of the hammer.

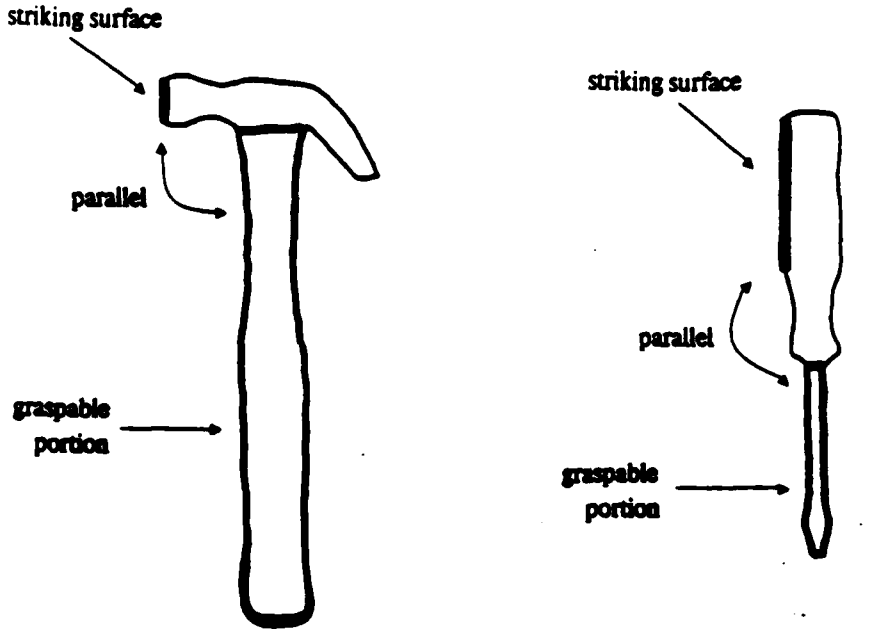


Figure 14. Function from structure. a. The structures of a hammer that contribute to its function. b. The functional description of a hammer mapped onto the structure of a screw driver.

We taught the system the function of a hammer by explicitly telling it the grasping and striking requirements, then showing it examples of various structures which are graspable, and that constitute striking surfaces. The advantage to having a functional description of a hammer is that it enables the program to improvise. Suppose the task is to drive some tacks but we do not have a suitable hammer. One solution is to use a screwdriver to pound them in. Our system can suggest this by using the functional definition we taught it (Figure 14). First, it notices that the blade of the screwdriver fits its structural description of a graspable part. Then it realizes that the side of the screwdriver's handle can be considered a striking surface. Since it is appropriately oriented with respect to the prospective handle, the system decides that the screwdriver successfully matches its description of a hammer.

6. Open Problems

We have a working vision system that generates quite rich descriptions of shapes, and a working learning system that can acquire complex concepts. However, there are a number of open problems. For instance, we have yet to work out the full ramifications of Gray coding. Also, the shape descriptions are currently restricted to two dimensions and only work well with objects composed of elongated pieces. We are working on surfaces

and volumetric representations [Brady, Ponce, Yuille, and Asada 1985, Ponce and Brady 1985] as well as à companion technique for round regions [Fleck 1985]. In addition, even in two dimensions, our representations are oriented to static rigid objects. Incorporating kinematics would be a first step toward representing form and function and would force us to confront issues such as causation, which we have largely finessed to date. With function and form, we can begin to think about planning for the *Mechanic's Mate*, as well as naive physics. Plainly, there is much to do.

7. Acknowledgements

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's Artificial Intelligence research is provided in part by the System Development Foundation, the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-80-C-0505, and the Office of Naval Research under contract number N00014-77-C-0389. We thank several people who have commented on the ideas presented in this paper, particularly Phil Agre, Steve Bagley, Bob Berwick, Ben DuBoulay, Alan Bundy, Margaret Fleck, Scott Heide, Boris Katz, David Kirsh, Tomás Lozano-Péres, John Mallery, Tom Mitchell, Sharon Salveter, and Patrick Winston.

References

Agre, Philip, [1985], "Routines", MIT Artificial Intelligence Laboratory, AIM-828.

Anderson, J. R, [1977], "Induction of augmented transition networks," *Cognitive Sci.*, 1.

Asada, Haruo and Michael Brady, [1984], "The curvature primal sketch", MIT Artificial Intelligence Laboratory, AIM-758.

Bagley, Steven C., [1984], "Using models and axes of symmetry to describe two-dimensional polygonal shapes" (SM Thesis), MIT Dept. Elec. Eng. and Comp. Sci.

Berwick, Robert C, [1985], *Localitiy principles and the acquisition of syntactic knowledge*, MIT Press (to appear), Cambridge, Mass.

Brady, Michael, [1983], "Criteria for representations of shape", *Human and machine vision*, A. Rosenfeld, B. Hope, and J. Beck (eds.), Academic Press.

Brady, Michael, and Haruo Asada, [1984], "Smoothed local symmetries and their implementation," *Int. J. Robotics Research*, 3 (3).

Brady, Michael, Philip Agre, David Braunegg, and Jonathan Connell, [1984], "The Mechanic's Mate", *ECAI 84: Advances in Artificial Intelligence*, T. O'Shea (ed.), Elsevier Science Publishers B.V., North-Holland, Amsterdam..

Brady, Michael, Jean Ponce, Alan Yuille, and Haruo Asada, [1985], "Describing surfaces," *Comp. Vis. Gr. and Im. Proc.*

Brooks, Rodney A, [1981], "Symbolic reasoning among 3-D models and 2-D images," *Artif. Intell.*, 17, 285 – 348.

Brooks, Rodney A., and Thomas O. Binford, [1980], "Representing and reasoning about partially specified scenes", *Proceedings, DARPA Image Understanding Workshop*, Lee S. Baumann (ed.), Science Applications Inc., 150 –156.

Chomsky, Noam, and Morris Halle, [1968], *The Sound Pattern of English*, Harper and Row, New York.

Connell, Jonathan H., [1985], "Learning shape descriptions: generating and generalizing models of visual objects" (SM Thesis), MIT Dept. of Elec. Eng. and Comp. Sci.

Davis, Randall, [1984], Diagnostic reasoning based on structure and behavior, MIT Artificial Intelligence Laboratory, AIM 739. (also in Artificial Intelligence (1985)).

Davison, L. D., and R. M. Gray, [1976], *Data Compression*, Dowden, Hutchinson and Ross.

Dietterich, T. G., and R. S. Michalski, [1981], "Inductive learning of structural descriptions," *Artif. Intell.*, **16**.

Doyle, Richard J., and Boris Katz, [1985], "Exploring the boundary between natural language and knowledge representation", MIT Artificial Intelligence Laboratory, Forthcoming AI Memo.

Duda, R. O., and P. E. Hart, [1973], *Pattern Classification and Scene Analysis*, Wiley, New York.

Fleck, Margaret, [1985], "Local rotational symmetries" (SM Thesis), MIT Dept. Elec. Eng. and Comp. Sci.

Freeman, P., and Allen Newell, [1971], "A model for functional reasoning in design", Proc. 2nd Int. Jt. Conf. Artif. Intell., London, UK.

Fu, K. S., [1971], *Pattern Recognition and Machine Learning*, Plenum Press, New York.

Fu, K. S., [1974], *Syntactic Methods in Pattern Recognition*, Academic Press, New York.

Fu, K. S., and J. T. Tou, [1974], *Learning Systems and Intelligent Robots*, Plenum Press, New York.

Gonzalez, R. C., and M. G. Thomason, [1978], *Syntactic Pattern Recognition: an Introduction*, Addison-Wesley, Reading, Ma..

Hamming, Richard W., [1980], *Coding and Information Theory*, Prentice-Hall, Englewood Cliffs, NJ.

Heide, S., [1984], "A hierarchical representation of shape" (SM Thesis), MIT Department of Mechanical Engineering.

Hilgard, E. R., and G. H. Bower, [1966], *Theories of Learning (3rd Ed.)*, Appleton-Century-Crofts, New York.

Katz, Boris, and Patrick H. Winston, [1983], "A two-way natural language interface", *Integrated Interactive Computing Systems* P. Degano and Erik Sandewall (eds.), North-Holland, Amsterdam.

Kenstowicz, Michael, and Charles Kisseberth, [1979], *Generative Phonology*, Academic Press, New York.

Lenat, Douglas B., and John Seely Brown, [1984], "Why AM and EURISKO appear to work," *Artif. Intell.*, **23**, 269 – 294.

Lindsay, P. H., and D. A. Norman, [1972], *Human Information Processing*, Academic Press, New York.

Marr, D., and H. K. Nishihara, [1978], "Representation and recognition of the spatial organisation of three dimensional shapes," *Proc. Roy. Soc. Lond. B*, **200**, 269 - 294.

Michalski, R. S., and R. L. Chilauski, [1980], "Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis," *Policy Analysis and Information Systems*, **4**.

Nilsson, N. J., [1965], *Learning Machines*, McGraw-Hill, New York.

Ponce, Jean, and Michael Brady, [1985], "Toward a surface primal sketch", *Three Dimensional Vision*, Takeo Kanade (ed.), Academic Press.

Vere, Steven A., [1977], "Relational production systems," *Artif. Intell.*, **8**.

Winston, Patrick H., [1980], "Learning and reasoning by analogy," *Comm. ACM*, **23**, 689 - 703.

Winston, Patrick H., [1981], "Learning new principles from precedents and exercises," *Artif. Intell.*, **19**, 321 - 350.

Winston, Patrick H., [1982], "Learning by augmenting rules and accumulating censors", MIT Artificial Intelligence Laboratory, AIM-678.

Winston, Patrick H., [1984], *Artificial Intelligence, 2nd. Ed.*, Addison-Wesley, Reading, Ma..

Winston, Patrick H., Thomas O. Binford, Boris Katz, and Michael Lowry, [1984], "Learning physical descriptions from functional definitions, examples, and precedents", *Robotics Research*, Michael Brady and Richard Paul (eds.), MIT Press, Cambridge, 117 - 135.

END

FILMED

9-85

DTIC